Attorney's Docket No.: 12481-004001

APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: REORGANIZING CONTENT OF AN ELECTRONIC DOCUMENT

APPLICANT: GREGORY C. SCHOHN, ADAM L. BERGER AND RICHARD D. ROMERO

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EI270011460US

I hereby certify under 37 CFR §1.10 that this correspondence is being deposited with the United States Postal Service as Express Mail Post Office to Addressee with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, Washington, D.C. 20231.

May 8, 2001
Date of Deposit

Samantha Bell
Signature

Samantha Bell
Typed or Printed Name of Person Signing Certificate

REORGANIZING CONTENT OF AN ELECTRONIC DOCUMENT

Background

This invention relates to reorganizing content in an electronic document.

Businesses and individuals that publish electronic documents—web pages, email, Microsoft Word documents, rich text files, for example—usually have a target platform in mind. Often the platform is a desktop computer with reasonable storage capacity, memory, bandwidth, and a reasonably large display. Because the electronic documents to be published are designed with this platform in mind, they often contain complex formatting information involving tables, frames, graphics, and navigational aids, all of which define how the document is to be rendered on a conventional computer monitor.

A user may wish to access such a document through a medium for which the document was not originally designed, for example, an Internet-enabled mobile phone, a personal digital assistant (PDA), or a handheld computer. These devices have limited screen size, resolution, and rendering capabilities and are typically unable to render such documents as they were originally designed. Internet-enabled mobile phones, for instance, usually can display only a few lines of text, and either grayscale, thumbnail-sized images or no images at all.

A different medium, which presents similar issues, is speech. People may access electronic documents by telephone, by dialing in to a service that uses speech synthesis to dictate the contents of the documents over the phone. *Voice Becoming the 'most powerful tool' on the Web*, Barbara Rose, Chicago Tribune, January 29, 2001. A dictated document does not express the complex layout information embedded in the original document.

Many Internet-enabled mobile devices restrict the maximum size of a document that they can render. For instance, most Internet-enabled phones that comply with the WAP (Wireless Application Protocol) standard support documents no larger than 2000 bytes. Even for those mobile devices (e.g., Pocket PC's and palm-based computers) that do not impose a strict size limit on documents, large source documents must be broken into smaller parts because transmitting long documents at once over slow wireless networks can try the patience of users. (For a similar reason, large documents are broken into smaller parts for the purposes of dictation.)

For example, the original hypertext markup language (HTML) of the web page shown in Figure 1A can be broken into nine subdocuments so that the page can be displayed in subdocuments on the screen of a phone as shown in Figure 1B. The top of the screen on the phone shows "(2/9)", indicating that the phone is displaying the second of nine subdocuments that made up the original HTML document.

Summary

Among other advantages of the invention, original documents can be reconstructed to enable a user of a low-capability device to view (or listen to) and use the documents effectively.

In general, in one aspect, the invention features a method that includes (a) receiving an electronic document represented by serial data that contains content of the document and defines an order in which respective portions of the content are to be performed, (b) analyzing the serial data of the electronic document, and (c) generating reorganization information for use in delivering the portions of the content, the reorganization information enabling performance in an order different from the order defined by the serial data.

Implementations of the invention may include one or more of the following features. The serial data representing the electronic document is expressed in a hypertext markup language such as HTML, PDF, postscript, SGML, Powerpoint, rich text, or unformatted text. In other examples, the serial data representing the electronic document is expressed in an electronic mail format that includes a header and a main body and the analyzing process includes determining the start of the main body. The content of the document includes at least one of the following: text, images, tables, frames, and headings. The order in which the respective portions of the content are to be performed may include a two-dimensional layout.

The reorganization information includes an identification of a relative importance of the respective portions of the content. The information includes an identification of a main block of text. The analyzing process includes finding an annotation inserted in the electronic document as a marker of the location of the main block of text. The reorganization information associates with each of the portions of the content a revised order for presentation. The reorganization information includes a hyperlink to be displayed near the beginning of the document, the hyperlink pointing to a portion of the content that appears later in the original order. The reorganization information causes an automatic redirection from the first portion of the content to a later portion of the content when the document is opened for performance. The hyperlink is included only if the location of the hyperlink in the document is separated by at least a predetermined distance from the location to which it points.

Performance of the portions of the content comprises presenting the portions on a display for viewing. The different order of performance enabled by the reorganization information is adapted for a display that has a more restricted performance capability than does the performance capability of the display for which the document was originally designed. The more restricted display is part of a mobile phone or personal digital assistant, and the display for which the document was originally designed comprises a desktop computer monitor. In some implementations, performance of the portions of the content comprises presenting the portions by speech synthesis.

The analysis includes identifying one of the portions as containing central content of the document. The generation of the reorganization information includes inserting a link from near the beginning of the first portion of the content to the beginning of the central content portion. The generation includes moving the central content portion to near the beginning of the document. The generation includes altering the document so that the central content portion appears first when the document is performed.

The analysis includes identifying portions of the document that should not be separated in generating the reorganization information. The portions that should not be separated include at least one of the following pairs: heading and text, image and caption, or paragraph and related paragraph. The analysis includes identifying portions of the document that should not be moved relative to other portions of the document. The analysis includes identifying portions of the document that should be moved relative to other portions in generating the reorganization information. The portions that should be moved comprise images or tables. The analysis includes identifying regions according to functions. The functions include navigation and content. The analysis includes converting the document to a tree format. The analysis includes blocking major regions of the document. The analyzing includes counting characters of text.

In general, in another aspect, the invention features a method comprising (a) at a server, receiving a request from a remote device for a portion of a document represented by serial data that contains content of the document and defines an order in which respective portions of the content are to be performed, in response to the request, and (b) returning at least one and fewer than all of the portions of the content using reorganization information that enables performance of the portions in an order different from the order defined by the serial data. Implementations of the invention include receiving other requests for portions of the content of the document different portions, and in response to the requests, returning other portions of the content using the reorganization information.

In general, in another aspect, the invention features a data structure stored on a medium and capable of configuring a machine to respond to requests for portions of a document that is represented by serial data that contains content of the document and defines an order in which respective portions of the content are to be performed, the data structure comprising reorganization information that enables performance of the portions in an order different from the order defined by the serial data. In implementations of the invention, the data structure may include the content, the data being expressed as a modified version of an original data structure that expressed the document. The modified version may include annotations.

In general, in another aspect, the invention features apparatus corresponding to the methods.

Other advantages and features will become apparent from the following description and from the claims.

Description

(Figures 1A, 6, 10, and 11 show web pages or portions of web pages..

Figures 1B, 7A, 7B, 7C, 8A, 8B, and 9 show mobile phone displays.

Figures 2, 3, 4, 5, and 12 show a block diagram.

Figures 13, 14, 15, and 16 show HTML-to-tree and tree-to-tree conversions.)

One difference between the capabilities of a desktop computer and of "non-traditional" media such as the phone is the display dimensionality. A desktop monitor supports a two-dimensional layout of information, whereas a phone's display or a sequence of dictated words appears "ticker-style"—as essentially a linear flow of information. (While technically the phone's display is two-dimensional, the amount of information that can be displayed is so small as to produce effectively a ticker-style impression.)

An automatic real-time content transformation system can transform a document meant for two-dimensional display into a ticker-style, one-dimensional stream of data, without producing a document that is inconvenient to read.

As shown in figure 2, part of the functionality of an automatic real-time content transformation system is to segment a source document 10 into smaller interrelated subdocuments 12, each of a size small enough to be processed by a target mobile device. More information about one way to perform the segmentation is set forth in United States patent application 09/745,289, filed 12/20/2000, and incorporated by reference.

A document targeted for display on a portable device is written in a markup language accepted by the device—sometimes HTML, but often another less commonly-used markup language such as WML, HDML, cHTML, or a proprietary language. Because different wireless data devices have different capabilities, a content creator writes a separate version not only for each target markup language but also for each target device. For instance, a mobile phone equipped with a Nokia microbrowser renders WML content differently from a phone equipped with an OpenWave microbrowser. Also, the content provider needs to understand how to identify the capabilities of the client device and how to create a document optimally formatted for that client.

As shown in figure 3, automatic, real-time content transformation 14 retargets a document 16 for optimal display on a target device 18.

As shown in figure 4, Internet-enabled wireless devices 20 communicate with the Internet through a gateway 22, which mediates between the wired and wireless worlds. When, for instance, the user of an Internet-enabled phone requests a web page, that request is transmitted to the wireless gateway, which in turn forwards the request to the origin server 24 on the Internet responsible (according to the DNS protocol) for the requested document. An automatic, real-time content transformation system can be deployed at various locations in the client-origin server channel, including the wireless gateway and the origin server.

As shown in figure 5, an automatic content transformation process 30 begins with a source document 32, for example, an HTML file, email, PDF, postscript, text, or XML file, among other formats, which is first processed by an adaptor 34 that converts the input to an internal representation (XML is one convenient format).

After the conversion is complete, the resulting intermediate document is sent to a sequence of content transformation modules 36, 38 to optimize it for display on the target device. The document is then reformatted to the markup language 40 appropriate for the target device and transmitted to that device. User preferences 42 and document caching 44 may be included in the system.

The resulting document is adapted to the type of client device requesting the document, so that, for example, a document intended for viewing on a PDA will be processed differently from one intended for use on an Internet-enabled phone. The transformation modules adapt not only to the type of device (handheld computer, PDA, phone, etc.) but also to the specific model of the device and to the wireless service provider. A Palm VII with the Palm.net wireless service has different behavior from a Handspring Visor with an Omnisky wireless modem, for instance, and a Samsung SCH8500 phone with Sprint PCS wireless web service has different characteristics than a Motorola StarTAC with AT&T wireless service.

Some simple examples of content transformation follow:

-Abbreviate dates such as “November 12, 1987” into a shorter form, like “11/12/87”.

-Replace phrases with common acronyms (so that “United States” becomes “U.S.”).

-Partition a large document into smaller parts, each small enough to be digestible by the target device.

-Shrink the size of each image to fit on the target device. Convert color images to black and white, for those devices that cannot render color images.

-Reorganize the document so it can more conveniently be accessed by someone using a non-traditional medium.

With respect to the final item of the list, figure 6 shows a typical commercial web page 50 having a complex, two-dimensional layout. Many people viewing this document on a traditional desktop computer display will first notice the story beginning "Britain Deploys." However, this story does not appear at the beginning of the source HTML document that underlies the displayed version. Rather, a number of advertisements, banners, forms, and navigation links precede the story in the source HTML, including A New York Times banner, the date and time, a link to "personalize your weather", a search form, and a set of navigation links, starting with "Classifieds," which are displayed on the left-hand side of the screen.

If the source HTML document were transmitted in its original order to a small-screen device, the user would have to navigate through a considerable amount of secondary content before reaching the breaking story. As shown in Figures 7A, 7B, and 7C, the main story, starting with "In its battle to contain...", would not appear until the middle 54 of the third sub-document (in figure 7C).

Users of small-screen devices typically prefer not to have to wade through information of secondary importance before reaching the information of interest to them.

As shown in figures 8A and 8B, one strategy for enabling a user to get to the information of interest quickly is to insert a link 56, at the beginning of the first subdocument, that links directly to the main content 58 that begins at subdocument 4 of 15).

Another strategy, shown in figure 9, is to reorder the original document, so that the main content appears first (in the source for the first subdocument)

A third approach is to provide an internal annotation to the subdocument containing the beginning of the main content and cause the display device to start directly at this subdocument when the user requests the document.

As shown in figure 10, another difficulty faced by those viewing documents using non-traditional media occurs when the original document includes, for example, a table 60 next to a body of related text 62. Such interrupting blocks can be identified and moved so they appear after, rather than in the midst of, the adjacent text. After rearrangement, the content becomes more accessible on linearly-formatted media like small-screen handsets.

In some implementations, one or more of the following operations (which can be thought of as subroutines) are applied to an input document (such as a hypertext document in HTML, XML, text, Microsoft Word, or another format). The output is a document whose content has been altered to allow for easier access through non-traditional media.

Functions to be performed by a restructuring algorithm

Annotate the beginning of the main content

The annotation is a single node inserted into a tree representation of the document, at the place where the system determines that the central content of the document begins There are two methods for determining where the main content begins:

1. Use, if present, a document author's annotation.

2. Calculate the location of the beginning of the main content using the algorithm described below.

Using this information, any of the three approaches mentioned earlier can be implemented: inserting a link from the beginning of the first subdocument to the beginning of the main content, reordering the document so the main content moves to the beginning of the first subdocument, or directing a user immediately to the beginning of the main content.

Annotate the scope (start and end) of atomic groups in the document

By "atomic group" is meant a group of sibling nodes within a document tree that should not be separated. For instance: (a) a headline should not be separated from the subsequent story, (b) a picture should not be separated from an accompanying caption, and a sequence of paragraphs comprising a body of text should not be separated from one another.

The purpose of identifying and annotating "atomic" blocks within the HTML code is to ensure that if content in a document is rearranged, the rearrangement does not violate the coherence of the content of the document.

Classify subtrees within the document tree as movable or not

Certain subtrees within a document tree—tables, table rows, table cells, and image maps—can be migrated within the document without disrupting (often improving, in fact) the narrative flow of the document. Elements that are not movable include paragraphs within a larger text block and images adjacent to a caption. Moving them would disrupt the narrative flow of the document.

Move elements that interrupt a body of text to locations outside the text body

As described earlier, punctuating a body of text with a related picture or table is a stylistic device often used by document authors and publishers. But such interruptions are often disruptive when the document must be conveyed in a linear manner. Therefore, these types of "accompanying" elements, when marked as movable, are demoted to the end of the text block.

Regions in the document are classified according to function

Regions in the documents are classified into one of a number of categories, listed in Table 1.

Template content	Narrative content that is generic or not related to the rest of the document (e.g. the copyright information, or information related to the revision history of the document.)
Default	The default or "catchall" category
Input/form-related	Elements related to transactions (forms, buttons, input text blocks, etc.)

Generic Navigation	A set of links with short labels whose purpose is to provide easy access to other documents.
Content Navigation	Navigational aids (links) which also contain information.
Content	Narrative content which appears to be unique to the document
Organizational Navigation	A set of intra-document links which point to parts of the current document as an aid in navigating the document.

Table 1: List of categories for nodes within a document tree.

Figure 11 shows a web page annotated with some of these categories, for illustration.

Example restructuring algorithm

Referring to figure 12, a step-by-step process 70 for content rearrangement includes the following steps.

Convert to tree format

The raw document 71, in an arbitrary format, is converted 72 to a common internal tree-based representation. The representation may be described using an emerging standard called the DOM (Document Object Model) markup language, Document Object Model (DOM) Level 3 Core Specification, Version 1.0 http://www.w3c.org, but other formats are possible. For documents in some markup languages there exist publicly-available tools for performing this conversion, The Tidy Project: http://www.w3.org/People/Raggett/tidy/, but for documents in other markup languages, the conversion routine must be created *de novo*.Figure 13 shows an example of a simple HTML source document 88 and a corresponding tree-based representation (with the subtree underneath the table node omitted for clarity) 90.

Block major regions

In the interest of clarity, long documents often include (implicitly or explicitly) information that demarcates major regions from one another. HTML authors, for instance, often use <hr> tags to separate regions; this tag typically appears as a thin line extending the entire horizontal span of the screen. HTML authors also sometimes use the <frame> tag to distinguish separate regions. In common word-processing formats such as Microsoft Word, the beginning of a new chapter or section serves to distinguish major regions. In presentation software such as Microsoft Powerpoint, separate slides represent different regions. Referring again to figure 12, for each major region explicitly demarcated in some way in the original document, the "Block major regions" component 74 identifies the region and inserts a BLOCK node in the document tree. The BLOCK node encapsulates the region, which exists as a subtree underneath the BLOCK node. Later processing will make use of this additional structural information in the document tree.

Figure 14 shows an example in which an HTML source document 92 having its first two regions demarcated by <hr> tags is represented by three Blocks in the tree representation 94.

Count Text

The count text step 76 counts the number of text characters within (and underneath) each node in the document tree. Although a document tree such as the one in figure 14 contains many characters, only those characters that will be displayed by a rendering agent (a web browser, for instance) are counted in this step. We'll subsequently refer to these text block characters as printable characters, distinguishing them from characters comprising element names ("img" and "bold" and "table", for instance).

Having counted printable characters, this step annotates each node with the number of printable characters within the subtree rooted at that node. We refer to this value as the text size of the node.

Mark Movable

The mark movable step 78 identifies movable elements—elements that can be moved within the tree. The actual moving of nodes occurs later, but we note that nodes typically will only be moved within their sibling set: the set of nodes which share the same parent in the tree. That is, a node generally is not promoted or demoted to a different level in the document tree.

Tables, table rows, table cells, image maps, and blocks generated from Block Major Regions are all movable. Individual paragraphs adjacent to other paragraphs are not movable, because moving one without the other could disrupt the correct ordering of text.

Aggregate

The aggregate step 80 encapsulates consecutive nodes in the tree that are acting as a functional unit. In this sense, it performs a function similar to Block Major Regions, except that the aggregate step operates at a finer level of granularity in the document tree.

This step achieves two main goals:

1. Protect groups of nodes within a document that are likely to have a similar purpose and should be kept together–groups of nodes that should not be rearranged, such as a sequence of paragraphs comprising a body of text.
2. Identify small nodes (typically but not exclusively textual) that act as labels for subsequent larger nodes, and protect against the later separation and rearrangement of these label/body pairs.

The aggregate step may itself be broken into three subroutines 81, 82, 83. Aggregate performs these three steps in sequence on each node in the document tree which has children.

Encapsulate Unmovable Blocks

The encapsulate unmovable blocks subroutine 81 establishes the following invariant in the document tree, maintained through the rest of the processing steps: *If one of a node's children is movable, then all the children are movable.*

To establish this invariant, this step finds contiguous sequences of unmovable nodes that are movable as a block, and encapsulates them inside a BLOCK, which is marked as movable.

```
EncapsulateUnmovable(Node n)

1. If n is not movable, then return
2. If none or all children of n are movable, then return n
3. Encapsulate (put underneath a BLOCK node) each contiguous sequence of
   unmovable child nodes of n
```

Figure 15 provides an example of a tree before 96 and after 98 the packaging of unmovable nodes.

Move Interrupting Blocks

As previously explained, an "interrupting block" is a set of elements that "interrupt" a body of textual content to provide an illustrative picture, supporting information, or in some cases a survey requesting feedback on the text. If not moved out of the way (by demoting them so they appear after, rather than during, the body of text), these interrupting blocks would disrupt the flow of the text within a linear presentation of the document.

A simple test for identifying interrupting blocks in an HTML document is to look for tables with the attribute align set to left or right. When found, the table is demoted so it appears after the last of its siblings that contains the adjacent text.

By performing this move interrupting blocks step on a node *n's* children immediately before the label attachment 83 of *n's* children, label attachment becomes much more accurate and easy to implement. Because labels and their bodies are determined by sizes of siblings, moving blocks that are to be moved anyway creates a single homogenous body instead of being separated across several disjoint? regions.

Find/Attach Labels

The find/attach labels subroutine 83 identifies nodes that act as labels for their successors. For instance, a headline acts as a label for the following story, and the two should not be segregated. The algorithm to accomplish this, shown below, begins by calculating a threshold value for each child of a node. That value is the geometric mean of the smallest text size and largest text size among the children. All siblings whose text size exceeds this threshold are labeled as LARGE, and the rest as small. The notion of LARGE and SMALL are thus relative to a set of siblings.

```
ClassifySiblingsByRelativeSize(Node n)

1. Classify each child of n as SMALL or LARGE as follows:

    a. Set min = minimum text size of all children of n

    b. Set max = maximum text size of all children of n

    c. Do for all children c of n:

        I. Set x = text size of c

        II. If $x < (\min * \max)^{1/2}$ then classify c as SMALL
            else classify c as LARGE

2. Encapsulate each consecutive sequence of SMALL children of n
   within a BLOCK, labeled as SMALL
```

```
3. Encapsulate each consecutive sequence of LARGE children of n
   within a BLOCK, labeled as LARGE
```

Steps 2 and 3 encapsulate similarly labeled siblings. Often this step captures many consecutive subtrees: for instance, a headline followed by a byline followed by a brief synopsis of the upcoming story. Connecting similarly labeled blocks ensures that the entire label and the entire block move as a unit, avoiding a separation of related blocks.

After these three steps, the following algorithm is used to attach labels to bodies.

```
AttachLabels(Node n)

1. Do for each consecutive pair of (SMALL, LARGE) siblings among the children of n:
    a. Let |x| = text size of node x
    b. Let |y| = text size of node y
    c. If |x| < |y|/3, then encapsulate (x,y) within a BLOCK
```

Step 1c is a heuristic (and the value 1/3 is a suggested value, which might not be optimal for certain classes of documents) designed to identify when a subtree is acting as a label to a subsequent block. The labeling strategy here is conservative, because the ramifications of mistakenly identifying a subtree as a label are small (merely that the subtree will never be separated from the subsequent block).

Classify

The classify step 84 classifies each node in the document tree into one of a fixed number of categories.The following table reiterates the list of the categories provided earlier and associates each category witha label, referred to in subsequent algorithms.

Template Content	TEMPLATE_CONTENT_BLOCK
Default:	DEFAULT_BLOCK

Input/Form Related:	FORM_BLOCK
Generic Navigation:	GENERIC_NAV_BLOCK
Content Navigation:	CONTENT_NAV_BLOCK
Content:	CONTENT_BLOCK
Organizational navigation:	ORG_NAV_BLOCK

The following algorithm contains an example classification procedure, designed for HTML documents. The return value is an integer priority, corresponding to the table of categories above.

```
int classify(Node n) {


        // A list of HTML tags which are input/form-related. Other markup
        // languages will have different tags.

        1.   formElementSet =

        {FORM, INPUT,BUTTON,TEXT_AREA,SELECT,OPTION,OPTGROUP,FIELDSET,LABEL};

        2.   if (formElementSet.contains(n)) return FORM_BLOCK;

        // There is no printable text within this subtree

        3.   if (n.textSize == 0) return DEFAULT_BLOCK;

        // Among all characters appearing in this subtree, what fraction

        // appears inside links and forms?

        4.   double inLinkRatio =

        (n.textSizeInLinks + n.textSizeInForms) / n.textSize;

        // The ratio of printable characters to links within this subtree

        5.   double textToLinkRatio = n.textSize / n.nLinks;
// This subtree contains links, a high percentage of characters
// inside links and forms, and a high percentage of same-site links.
// Note: n.nInDocLinks = # of links within the subtree rooted
// at node n which point elsewhere in the same site.
6.  if (n.nLinks > 0 &&
      (inLinkRatio > 1/2 && (n.nInDocLinks / n.nLinks > 2/3)))
    return ORG_NAV_BLOCK;
// Test for content / template content
7.  if (inLinkRatio < 1/2 && (n.nLinks == 0 || textToLinkRatio > 50))
    if (n contains the word "copyright") return TEMPLATE_CONTENT_BLOCK;
    return CONTENT_BLOCK;
    }
// There are no links within this subtree, or the ratio of text
// to links is very high
8.  if (n.nLinks == 0 || textToLinkRatio > 30)
    return CONTENT_NAV_BLOCK;
// base case
9.  return GENERIC_NAV_BLOCK;
}
```

Step 7 contains an overly simply heuristic–check for the word “copyright”–for determining whether a content block is actually template content. In practice, a more reliable test for template content would involve applying a text classification procedure, such as the Naïve Bayes classifier, to the task of distinguishing the two categories, Some Naïve Bayes classifier reference. Applying a machine-learning technique like Naïve

Bayes requires a large collection of text blocks, each annotated with the correct label (CONTENT or TEMPLATE CONTENT), so the algorithm can "learn" to distinguish the two categories.

In practice, the above heuristic works well for most HTML documents, especially those from websites with large, complicated pages that need to be distilled for lightweight devices. The algorithm above is also independent of the language or words that are being used. In addition to being portable to other languages, this also technique is also very fast compared to one that would need to do content analysis.

Node comparison routine

We now arrive at the final step of the document rearrangement process. Before describing the individual steps (insert link to main content 85, reorder document 86, and insert marker at main content 87), we introduce a node-comparison routine shared among these steps.

The comparison function places an ordering on the nodes by their classification. As one would expect, the CONTENT classification has a high priority, though not as high as ORG_NAV. Organizational navigational content is by definition a block that must precede the content because the hyperlinks within it point to places further down the tree. For instance, the links under "*In this story:*" on CNN's story pages, or the links in the left column on http://espn.go.com/jobs act as a table of contents to the main content and could be quite useful to a user of a lightweight device.

In cases where two nodes are both labeled as CONTENT blocks, the "block density" is used to break the tie. To define block density, we first define the Squared Block Size (SBS):

```
For all terminal blocks nodes:
        if (node is CONTENT) SBS = $textsize^2$
        Else SBS = 0
```

For all other nodes:

SBS = Sum of all childrens' SBS values

Block nodes are those that are elements that are considered block elements by the HTML specification. These elements can be thought of as not being able to occur on the same line with any other element. Examples are P, CENTER, DIV, BLOCKQUOTE, TD, etc. A terminal block is one that has no blocks underneath it.

The block density can be defined as:

D(a) = SBS of a / (# of terminal movable blocks under a)

More specifically, the "density" is the average SBS value for the terminal movable blocks under it. If there are two subtrees *a* and *b*, each containing 100 characters, but subtree *a*'s characters all appear within a single node whereas *b*'s characters are interspersed among many nodes, then subtree *a* is denser. The intuition here is that denser nodes are likely more descriptive (because their blocks are longer).

The comparison algorithm is therefore:

```
CompareSiblings (Node a, Node b)

    1. If (type of a != type of b) then return node of higher priority

    2. Return whichever node has the higher D-value
```

We now describe the three final steps in the reorganization process.

Insert Link to Main Content 85

The following algorithm locates the "main" CONTENT block in the document, and inserts a link from the beginning of the document to this block.

InsertLink

```
1. Set n = node at root of document tree
2. while (n is not a terminal cell AND n.textsize > K)
3.   if n has CONTENT block descendents then
4.     Set n = child CONTENT block with the highest D-value
5.   else break
6. // iterate back up the tree
7. while (n's previous sibling == LABEL OR n has no previous sibling)
8.   n = n's parent
9. If there are more than M printable characters between the start of the document and n, then insert a link from
   top of document to node n
```

In other words, walk down the tree while the nodes have at least K printable characters until a terminal cell is reached; at each level of the tree traversing the "best" content block. (The value of K is an adjustable parameter. In one implementation, K was set to 400.) Once this is found, make sure that a label would not appear right before the block in an in-order traversal (since that label would likely be part of the main content).

The value of *M* dictates how far from the beginning of the document the detected main content must reside before the algorithm will bother to insert a "jump to main content" link at the top of the first subdocument. It would make little sense, for example, to insert a "jump to main content" link when the main content is only three lines from the start of the transformed document.

Call *InsertLink*, but replace 11 with code to tell the system to start at *n* when the user accesses the document.

Reorder

The reorder step 86recursively sorts the children of each node in the document tree.

Before explaining the sorting procedure, we require a definition:

> *A node in a document tree is protected if its children are not movable, or if the subtree rooted at that node contains fewer than some predetermined number of characters N, or if the node was marked a label or body of a label earlier.*

"Protected" nodes are nodes into which the recursive sorting algorithm does not descend. In one implementation, *N* was set to 400.

Recall that the Encapsulate Unmovable Blocks step has previously ensured that either all or none of a node's children are movable.

The end result of the sorting procedure is a transformed tree in which the following holds: if a set of sibling nodes is movable, these nodes are ordered (from left to right) by decreasing likelihood of containing content.
Figure 17 shows a simple example of the sorting process applied to three children of a "document" node.

The sorting procedure is straightforward. Each node in the tree already has been assigned a category (in the Classify step). Nodes are sorted according to the ranking of categories given in Figure 15. If the two nodes belong to the same category, the sorting algorithm breaks the tie by preferring the node that contains a "denser" presentation of information.

Building a recursive node sorting algorithm on top of this node-comparison routine is straightforward:

RecursiveSort

```
1. Set n = root node of of document tree
2. If n is not protected, then
    a. Sort children of n with CompareSiblings algorithm
    b. Call RecursiveSort on each child of n
```

3. Return *n*

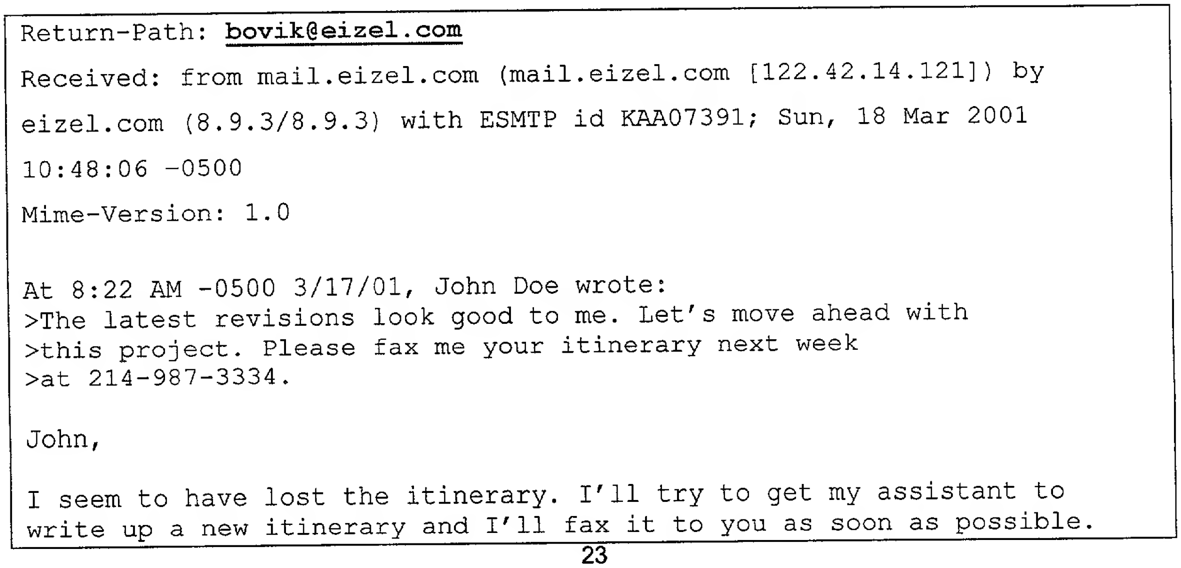
Other implementations are within the scope of the following claims. (let me know if there are any useful alternatives to the techniques discussed in detail earlier that ought to be mentioned.)

The above algorithms calculate the location of the beginning of the main content in a hypertext document. In some cases, this work isn't required. For instance, the author of a hypertext document may insert an annotation into the document to indicate where the main content begins.

Email

The above discussion relates generically to hypertext documents, such as web pages and corporate intranet documents. Similar principles can be applied to hypertext-encoded email messages. In addition, email documents, both hypertext-encoded and non-hypertext encoded have some particular characteristics, not found in general hypertext documents, that an automatic content rearrangement system can exploit for the purpose of reorganization. These characteristics present the opportunity for document reordering and prioritization for purposes of presentation.

The following figure is an example of a rather "generic" email.



```
Return-Path: bovik@eizel.com
Received: from mail.eizel.com (mail.eizel.com [122.42.14.121]) by
eizel.com (8.9.3/8.9.3) with ESMTP id KAA07391; Sun, 18 Mar 2001
10:48:06 -0500
Mime-Version: 1.0

At 8:22 AM -0500 3/17/01, John Doe wrote:
>The latest revisions look good to me. Let's move ahead with
>this project. Please fax me your itinerary next week
>at 214-987-3334.

John,

I seem to have lost the itinerary. I'll try to get my assistant to
write up a new itinerary and I'll fax it to you as soon as possible.
```

```
Harry
```

The categories described earlier are not well suited for email documents. For email, the following categories are more appropriate:

HEADER_BLOCK: The initial set of lines, beginning with a token which ends in a colon.

INCLUDED_MESSAGE: An email or part thereof prefaced by ">" or "|" or another indicative character. This also includes an optional preceding line(s), containing text such as "`At [time], [person] wrote:`"

MAIN_BODY: The content of the message itself.

Standard parsing algorithms can classify a line from an email, with high accuracy, into one of these categories. (In one example, the parser will have at least a one-line lookahead buffer.)

The main content, in this case, will be at the beginning of the main body. In the example above, this is the line which reads "`John,`". Given this classification, an automatic document restructuring system can apply the same policies--reorder the content, start at the main content, or insert a link to the main content--to an email document.